Multimedia Systems (MS)

Multi - numerous or multiple.

media - means of communication

Multimedia - Computer controlled integration of text, graphics, drawings, and any other media - where every type of info can be represented, stored, transmitted and processed digitally.

Perception media

→ Visual media - Text, graphics

→ Auditory media - Music, sound, voice

Motion Picture Camera - 1887 - Thomas Alva Edison

Size: a. 512 x 512 gray scale = $\frac{1}{4}$ MB

$= 512 \times 512 \times 8$

$= 20,97,152/8$

$= 262,144$

$= \frac{262,144}{1\,MB}$

$= \frac{1}{4}$ MB

$= \frac{1}{4}(1024)(1024)$

$= 262,144$

$1\,MB = 1,048,576$

Audio :-

→ Sampling rate in hertz.

→ Also described in dimensions.

→ 1D dimensions says number of channels that are contained in signal mono (one channel), stereo (two channels)

- 1 Minute of Mono CD quality - 5MB data
- 1 Minute of Stereo CD quality - 10 MB

→ Audio signals need to be compressed.

Video :-

→ Analog video is usually captured by a video camera

Monochrome video → PAL (Programming Array Logic) for colour video.
↳ (Phase Alternate line)

2. High Defn video

→ Image has ; width, height, pixel depth

→ video also have frames per second (fps).

Desirable features for Multimedia system :-

→ Very high processing power.

→ Multimedia capable file system

→ Special Hardware / Software needed

→ Data Representations.
→ Efficient and High I/O.
→ Special Operating System
→ Storage and Memory.
→ Network support.
→ Software Tools

Components of a multimedia system

→ Multimedia content creation
→ Compression & media encoding.
→ Distribution via networks

Challenges :-

→ How to represent data
→ Create Multimedia distribution
→ Distribute } & Multimedia { Effecively
→ Represent
→ Store

13/01/22 [HPC]

Time of start & finish op

$$\frac{\text{Seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{Seconds}}{\text{cycle}}$$

CPU execution time = CPU clock cycles for a program × clock cycle time

① Performance = $\frac{\text{CPU clock cycles}}{\text{clock rate}}$

Eq

Performance eq - ②

CPU execution time = Instruction count for a program × average CPI × Clock cycle

→ Computer A — clock cycle time = 250ps
CPI = 2.0

→ Computer B — 500ps = clock cycle
CPI = 1.2

CPS = clock cycle per Instruction

$$\text{CPU clock cycles}_A = \text{Instruction} \times \text{average CPI}$$

$$= 1 \times 2.0$$

$$\text{CPU clo cycles}_B = 1 \times 1.2$$

$$\text{CPU time}_A = \text{CPU clock cycles}_A \times \text{clock cycle time}$$

$$= 1 \times 2.0 \times 250\text{ps} = 500 \times I_B$$

$$\text{Performance} = \frac{1}{\text{Execution time}}$$

$$\text{CPU time}_B = 1 \times 1.2 \times 500\text{ps} = 600 \times I_{ps}$$

$$\frac{\text{CPU Performance}_A}{\text{CPU Performance}_B} = \frac{\text{Execution Time}_B}{\text{Execution Time}_A} = \frac{600 \times 2ps}{500 \times 2ps}$$

$$= 1.2$$

$\rightarrow$ A is 1.2 times fast than B

CPU clock cycles = Instruction count for a program $\times$ average CPI

$$\text{CPU clock cycle} = \sum_{i=1}^{n} CPI_i \times C_i$$

Amdahl's Law

18/1/22

0 - 9 [48 - 57]

A - Z [65 - 90]

a - z [97 - 122]

2 | 65
2 | 32 - 1
2 | 16 - 0
2 | 8 - 0
2 | 4 - 0
2 | 2 - 0
1 - 0

H E L L O
| | | | |
72 69 76 76 79

↓

2 | 72
2 | 36 - 0
2 | 18 - 0
3 | 9 - 0
3 | 3 - 0
1 - 0

2 | 72
2 | 36 - 0
2 | 18 - 0
2 | 9 - 0
2 | 4 - 1
2 | 2 - 0
1 - 0

64 + 8

H - 1001000

=

E - 69

2 | 69
2 | 34 - 1
2 | 17 - 0
2 | 8 - 1
2 | 4 - 0
2 | 2 - 0
1 - 0

= E - 1000101

L - 76.

= 1001100.

2 | 76
2 | 38 - 0
2 | 19 - 0
2 | 9 - 1
2 | 4 - 1
2 | 2 - 0
1 - 0

O

O = 79

= 1001111

2 | 79
2 | 39 - 1
2 | 19 - 1
2 | 9 - 1
2 | 4 - 1
2 | 2 - 0
1 - 0

= 1001000 1000101 1001100 1001100 1001111

31 bytes

59 bytes

Extended ASCII → 1 bit extra than 7 bit. So, 8 bits upto 256 characters.

Unicode:

→ from 0 to 127 it is ASCII

→ Covers 129 modern & historic scripts

→ Can represent more than 120,000 characters

→ UTF-8, UTF-16, UTF-32,
↓
defines 8 bit per character.

→ Larger no. of bits available than ASCII.

Two major problems:

→ Lot of lost valuable space.

→ ~~If after 8 zeroes~~ Ones, after 8 zeroes continuously would not be considered and interpreta would think as a null character (or) end.

110 → To first byte

10 → To start of second byte

Works for 2^{11} → 2048 bytes

→ Now, First byte is added!

Reason 110xxxxx → 5

10xxxxxx → 6

UTF-8 Separate

[1110 xxxx] [10 xx xxxx] [10 xx xxxx]

upto 2^{16} characters

more than 2^{16} ?

Add 1 in first byte. → 21

[11110xxx] [10 xxxxxx]

[10 xxxxxx] [10 xxxxxx]

(8) How the Devanagari character

क with code point 2325 can be

represented in: UTF-8, -16, -32

2325

→ 100100101 → 12 bits

6

8

utf-8

1110xxxx 10xxxxxx 10xxxxxx

all zeroes 100101 010101

→ utf-8

00001001 00101010 → utf-16

→ utf-32

0000

$11 - 10011011$ → UTF-8
9 bits

67

20/1/22.

Digital Image Representation:

→ Most common form to represent raster images is bitmap.
[Given bits maps to a specific color]

Bit depth: No. of bits required to represent each pixel (1 bit)

→ 1 bit images are on-off bits
[simplest type of image] [like

→ A 640×480 monochrome image requires 38.4 KB of storage.

→ (8-bit) Level Images:

→ Gray value between 0 to 255.
→ Each pixel is represented by a single byte

→ Set of 1-bit planes, a bit is turned on if the image pixel has a non-zero value at that bit level.

11100110 → 230

→ Each bitplane can have 0 or 1, but together it makes from 0 to 255.

640×480 gray scale image
= 307,200 = 300KB

24-bit

→ Each pixel is represented by 3 bytes, usually representing RGB

Resolution:
→ How finely a device display graphics with pixel

Ways of measuring resolution:

→ dpi (Dots per inch)
Printers & scanners have higher dpi.

→ ppi (Pixels per inch)
→ Resolution of bitmap images.

→ Bitmap with 72 PPI = 72 x 72 pixels
= 5184 pixels

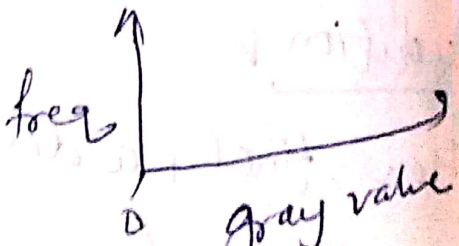
15/1/22

→ For an 8-bit image, image file can store in the file header info just what 8-bit values for R, G, B correspond to each index.

→ The LUT is often called as palette
↓
Look-up-table.

Color Histogram:

→ Carefully choose just which colors to represent in image.



Color Histogram

Color picker:

LUT = 256 x

136

230

8x

5 R's — 30 ×5 = 150

→ 40 x

760
736
996

800
260
10

1060

744
260
1004

25/1/22

→ High bit-depth is used in satellite imaging
Ex:- Types of growth etc.

→ Extra colours are required. medical
→ Multi-spectral (more than 3 colors)
→ Hyperspectral (224 colors) - satellite
→ Can be obtained by invisible light (infrared, ultraviolet)

Popular Image File formats :-

- 8 bit GIF : 8 bit colour images.
- JPEG
- PNG → lossless image format.
- TIFF → Flexible File Format.
- EXIF – allows addition of image metadata.
- PS & PDF - vector based language.

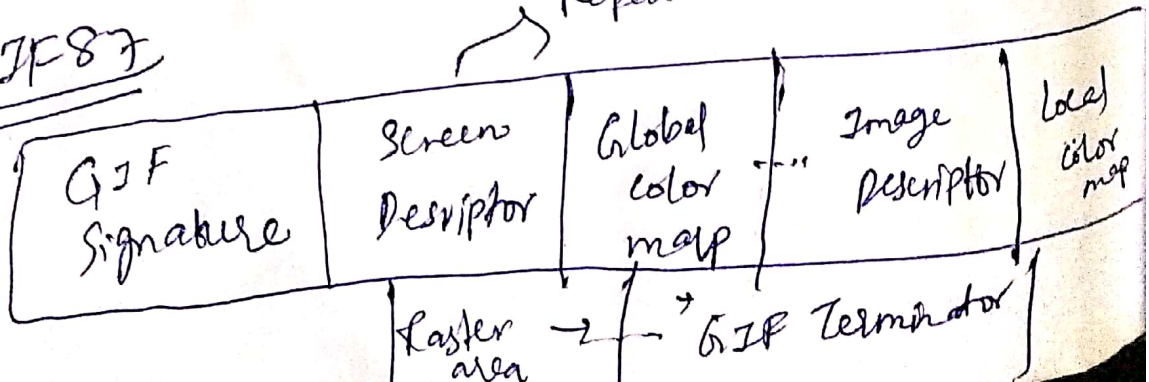
GIF (Graphic Interchange Format) :-

→ Limited to 8-bit (256) color images
→ Supports interlacing - successive display of pixels in widely-spaced rows by a 4-pass display process

Pass:
1 – 8th line
2 – 4th line
3 – 2nd line
4 – All other lines

→ GIF87a - original
→ GIF89a - Animations

GIF87



Signature - 6 bytes - Says that it is a GIF file.
GIF 87a - 7 byte set of flags
(Screen Descriptor)

Each image definition - own look up table
- A local color map
- Map 8 bits into 24-bit RGB

GIF Image descriptor
Contains details like Image left, Image Top, width, height, formatted, Global/local color map.

=

27/01/22 Raster area :

→ Series of pixel color index values that make up the image.

8 - 4 - 2 - 1

→ GIF four-pass interlace display row order

GIF Terminator

Character 0X3B hex or ';' is found after an image, then termination of GIF image file occurs.

Next Image File Format [JPEG]+

→ current standard for image expert compression

→ Joint photographic Experts Group (JPEG)

→ Many changes within few pixels, high image segment is referred as, Spatial frequency.

→ Divide by some large integer / truncate

→ lossy compression

→ Desired level of quality, compression ratio. [Input / output]

→ Usual default quality factor Q = 75

Original = 5.6% [Q = 75]

= 2.3% [Q = 10]

⇒ PNG [Portable Network Graphics]

→ Support for 16 bits per pixel in each color channel, 48-bit color.

→ 2D fashion — 7 passes through each 8×8 block.

- Supports lossy & lossless compression with performance better than GIF.

lossy → Can't construct the original image

lossless → Can construct ——————

TIFF

[Tagged Image File Format].

→ Aldus Corporation in 1980s

→ Additional info such as "tags", to provide great flexibility.

→ Tag Ex: Format Signifier: Type of compression

1 bit, gray scale, 8-bit color, 24-bit RGB,

↓ ↓

Binary 8-bit

→ Initially lossless, but can add JPEG tag for lossy compression.

EXIF: ["Exchange Image File]

→ Format for digital cameras

→ Does the image metadata recording.

[exposure, light source, flash, ...)

→ can be used in printers due to its meta data.

→ Incorporated in JPEG Software in digital cameras.

PS and PDF :

- Postscript.
  - → Important lang for typesetting.
  - → printers have postscript interpreter built into them
  - → Vector-based picture language rather than pixel based
  - → Includes text as well as vector/structured graphics.
  - → Stored as ASCII.

Portable Document Format (PDF) :

- LZW compression [2:1, 3:1]

WMF : Vector graphics file for microsoft

BMP : (Bitmap Image Files)

Major file format for Microsoft windows

Netpbm format : PPM (Portable Pixmap)

PGM (Graymap) PBM (Bitmap) - below

→ to opensource Netpbm formats.

* Common in Linux / Unix

11/2/22

RGB Model :-

→ Unit cube defined on R, G, B, axis

→ (0,0,0) → black.

(1,1,1) → white.

→ vertices are primary colors.

→ Shades of gray are represented along main diagonal.

$C(\lambda) = R(R) + G(G) + B(B)$

R, G, B are assigned values in range [0,1]

Each point as weighted sum of vectors.

CMY / CMYK Model [Subtractive model]

(Cyan, Magenta, Yellow)

- Used for hardcopy models (devices)

In RGB,

White is combination of primary lights,

black is absence of light

In CMY.

White is natural color.

Black is full combination of colors

Links.

→ CMY to RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

→ RGB to CMY [0, 1]

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

[0, 255]

$$\begin{bmatrix} 255 \\ 255 \\ 255 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

CMYK :

→ Black ink in addition.

→ Cost effective

→ Speed in drying

Undercolor removal :-

$k = \min\{C, M, Y\}$

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} \Rightarrow \begin{bmatrix} C-k \\ M-k \\ Y-k \end{bmatrix}$$ } new specificati of inks

K - amount of black

- Complimentary colors are opposite vertices
- Primary colors are vertices

3/2/22

Analog video standards

→ NTSC Video

- National Television System committee,
- Most of North America & Japan.
- 525 scan lines per frame at 30 fps

PAL Video :-

- (Phase Alternating Line) - German scientists
- 625 scan lines per frame, 25 fps
- Chroma signals has alternate signs (+U and -V) in successive scan lines

SECAM Video

→ Systeme Electronique Couleur Avec Memoire.

- Invented by French
- 625 scan lines per frame, 25 frames per second, 4:3

→ 4.25 MHz – U
4.41 MHz – V.

→ Sent Alternately.

U V B.

= [Types of video Signals]

Composite video + [Analog waveform]

- Baseband video (or) RCA video.
- Both chrominance (color) Luminance (brightness)

S-video (Super video) = (Y/C Video)

- Luminance & chrominance signal Sent separately
- Luminance (Y) – light
- Color (Chrominance signal – (C)

U & V Sig

- Reduces interference.
- Improves superior quality.

Component video :-

→ Y, U, V components separate
→ More Bandwith
→ Best visual quality
→ NO Artifact [Error] - Noise

(Red, Green, Blue Wires

Digital video

→ Storing video like in memory, noise removal.
→ Direct access, nonlinear video editing Simple
→ Repeated recording.
→ Ease of encryption & better tolerance to noise.

→ Television works by sending scan line info through YUV format.

→ In digital video, Subsampling can be done by reducing the no of bits used for color channels on avg.

→ In analog, half bandwidth to chrominance & Subsampling is done other half to luminance

→ Circles represent pixel info.

- Digital video

→ 1 byte is reserved for luminance

→ x at pixel position suggests that we also store chrominance components for this position.

8 bits - ~~4:4:4~~

$\frac{24+8}{2} = \frac{32}{2} = 16$

$\frac{24+8+8+8+(24)^{x}}{4} = \frac{24(3)}{4} = 18$

6

8 4 4

15/12/22

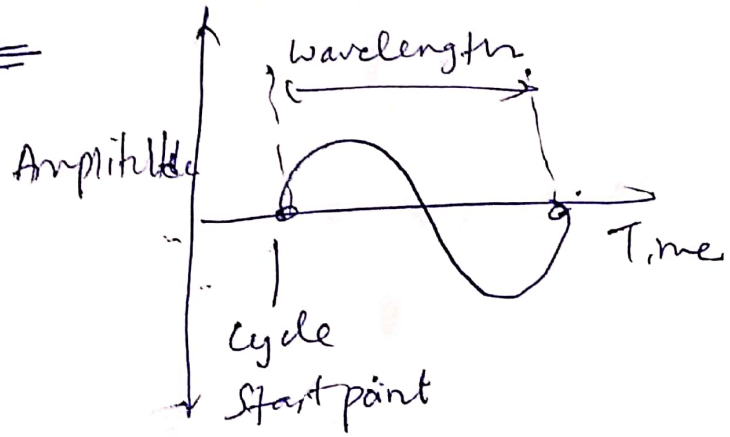
Multimedia Imaging Libraries

- PyMedia
- Gstreamer
- Pyglet
- Pygame
- Sprite

262144

Input Image - convert ('L')

Conversion to Gray scale



Only 1 independent

So, 1D Signal

→ Sampling → Measuring the quantity in evenly spaced intervals

Rate at which it is performed

↳ Sampling rate, Sampling freq

$$\text{Sampling period} = \frac{1}{\text{Sampling rate}}$$

20 Hz to 20 kHz

→ Sampling freq is increased after sampling

→ To preserve full info in signal, we sample at twice the maximum freq of signal. Nyquist rate.

→ When we will convert back to continuous signal (if sampled at freq lower than nyquist rate), aliasing occurs.

→ Aliasing is presence of unwanted components in reconstructed signal.

Quantization :-

→ Transforming a sampled analog signal to digital signal, which has a discrete set of values.

→ Levels increase, Quantization error decrease.

Quantization error :- Diff between analog signal & closest available digital value at each sampling instant.

~~8 bit~~

→ Typical uniform quantization rates are 8-bit into 256 levels
16 bit into 65,536 levels

---

max ⇒ One half of discretization step error
in quantization

Multi-part form data handling -

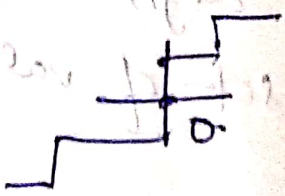
=

17/2/22

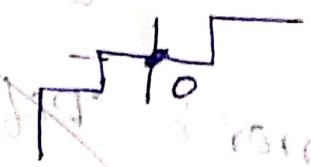
4 bit quantization ; 16 bit levels

Mid-Rise & Mid-Thread type Quantization

Mid-Rise → Levels are even in number
origin in middle of rising part



→ Mid-tread



odd levels
of quantization.

Between input value & quantized
called quantization error.

Non-uniform quantization

→ Non-linear .. Characteristics
→ step size is not constant.
Crest factor = peak signal / rms value.
→ Achieved using companding.

Companding +

Compression & Expanding

Input → Compressor → Uniform quantizer → Compressed signal

Uniformly q- ← Expander
version of original

Com

compr

→ Voice in → Encoder → digital
Comp

Vout

expansion

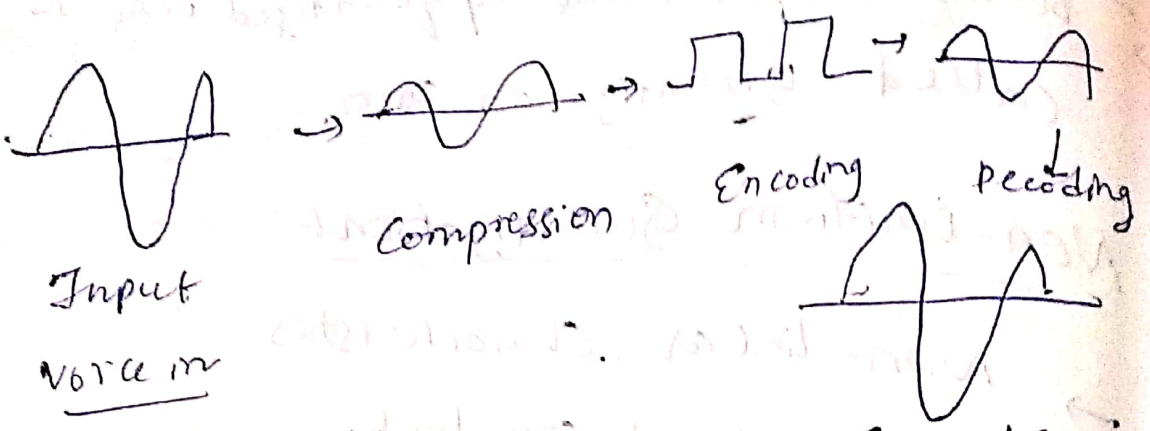
Vin

comp

-Vout.

← Decoder

Expander



μ law & A-law companding

↓

Popular

$$\rightarrow \quad y = \frac{\text{sign}(s)}{\ln(1+\mu)} \ln\left\{1 + \mu \left|\frac{s}{s_p}\right|\right\} \qquad \left|\frac{s}{s_p}\right| \leq 1$$

$\mu = 255$

$\mu \uparrow$ non-linearity $\uparrow$. compression $\uparrow$.

A-law

$$y = \begin{cases} \dfrac{A}{1+\ln A}\left(\dfrac{s}{s_p}\right) \\ \dfrac{\text{sign}(s)}{1+\ln A}\left[1 + \ln A\left[\dfrac{s}{s_p}\right]\right], \quad \dfrac{1}{A} \leq \left|\dfrac{s}{s_p}\right| \end{cases}$$

$$\text{where sign}(s) = \begin{cases} 1 & \text{if } s \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

$A = 100$

→ S = actual signal value

S_p = peak value

Modulation = varying one or more parameters of a carrier signal in accordance.

→ **Digital modulation technique** → Pulse Code Modulation

→ creating digital signal from analog using sampling & quantization

Formal name is pulse code

PCM

Input analog Speech signal → Bandlimiting filter → μ law or A law compressor → Uniform quanti / Linear PCM (compress)

→ Transmiss

Digital to analog → μ law (or A law expander) ← Low pass filter ← Output analog Speech signal

22/2/22

SU - 32
SD - 32

32 32 32
SU, SU, SU, 4

$= 100$ → $\hat{f}_n = \left[\frac{1}{2}(f_{n-1} + f_{n-2})\right]$

64
32
96

$e_n = f_n - \hat{f}_n$.

$f_0 = 21$
$f_1 = 21$
$f_2 = 22$
$f_3 = 27$
$f_4 = 25$
$f_5 = 22$

$\hat{f}_2 = \frac{1}{2}[f_1 + f_0)]$

$= \frac{1}{2}(42) = 21$

$\therefore e_n = f_n - \hat{f}_n$

$e_2 = 22 - 21 = 1$.

$e_2 = 1$

$\hat{f}_3 = \frac{1}{2}[f_2 + f_1]$

$= \frac{1}{2}(43) = [21.5] = 21$

$e_3 = f_3 - \hat{f}_3 = 27 - 21 = 6$ $\quad e_3 = 6$

$\hat{f}_4 = \frac{1}{2}[f_3 + f_2) = \frac{1}{2}(49) = 24.5 = 24$

$e_4 = 25 - 24 = 1$ $\quad e_4 = 1$

$\hat{f}_5 = \frac{1}{2}(f_4 + f_3) = 26$

$e_5 = f_5 - \hat{f}_5 = 22 - 26 = -4$

$e_5 = -4$

Differential Pulse Code Modulation (DPCM)

is exactly the same as Predictive Coding, Predictive coding except that it has a quantizer step.

Original f_n

Predicted $\hat{f}_n$

Quantized reconstructed $\tilde{f}_n$

$e_n = f_n - \hat{f}_n$

$\tilde{e}_n = Q[e_n]$

$f_n = \tilde{f}_n$

Quantiz noise

transmit codeword $(\tilde{e}_n)$

reconstruct $\tilde{f}_n = \hat{f}_n + \tilde{e}_n$

→ $\tilde{e}_n$ [Quantized error values are produced using entropy coding ;]

→ Distortion is average squared error

$$\frac{\sum_{n=1}^{N} (\tilde{f}_n - f_n)^2}{N}$$

Quantization noise $f_n - \tilde{f}_n$ is tilled

$\hat{f}_n = \text{trunc}\left[\frac{1}{2}(\tilde{f}_{n-1} + \tilde{f}_{n-2})/2\right]$

$e_n = f_n - \hat{f}_n$

$$\tilde{e_n} = Q[e_n] = 16 \times \text{trunc}\left\{\left(255 + \text{sign}(e_n)\right)/16\right\}$$
$$= 256 + 8$$

$$\tilde{f_n} = \hat{f_n} + \tilde{e_n}$$

Delta modulation (DM):

- Simplified version of DPCM often used as a quick analog-to-digital converter

$$\hat{f_n} = \hat{f}_{n-1}$$

$$e_n = f_n - \hat{f_n} = f_n - \hat{f}_{n-1}$$

$$\tilde{e_n} = \begin{cases} +k & \text{if } e_n > 0, \\ -k & \text{otherwise} \end{cases}$$

k is constant

$$\tilde{f_n} = \hat{f_n} + \tilde{e_n}$$

→ Not useful when rapidly changing

→ Only useful for constant.

Adaptive DM:

→ Changing step size

→ The step size is adaptively

22022022

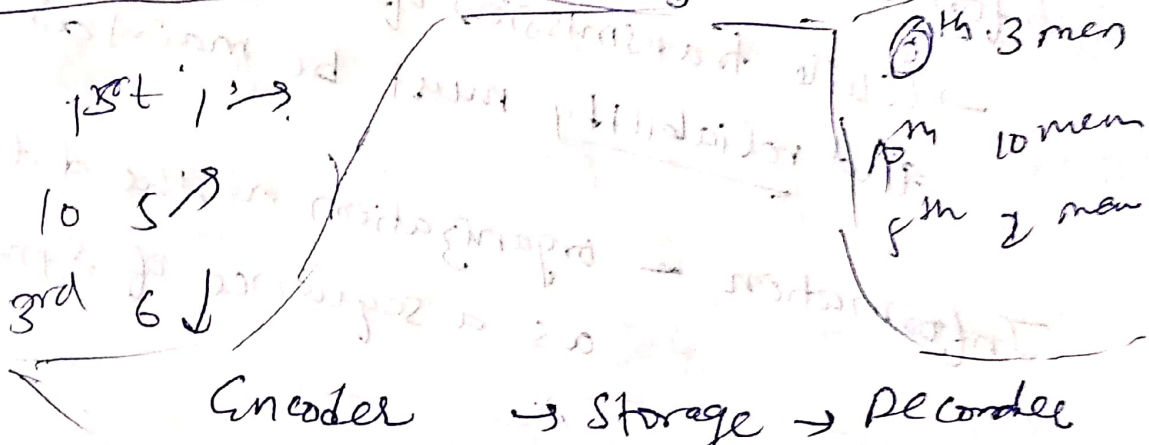
- Ambigram eg Palindrome

MP3 - MPEG Audio Layer-3

↓

Moving Pictures Experts groups

Scheduled quiz, ~~Tuesday~~ Thursday, 1st ~~Feb~~ March, 4-5pm



Encoder → Storage → Decoder

(Compression) (Decompressing)

name → End Exam

4/2/22

→ Output of encoder codes or codewords.

→ If compression/decompression process produce no loss, compression scheme is called lossless.

Compression Ratio:-

To represent data before compr — B_0

" data after compr — B_1

→ We would require any codec to have high compression ratio

Information Theory:-

→ While transmission of data, efficiency and reliability must be maintained.

Information — organization in the data as a sequence of symbols

---

1 pixel = 8 bits

∴ ? = 80,000 bits

10,000 pixels

∴ Each symbol / pixel = 8 bits

— Total possible $2^8 = 256$ possible gray level symbols.

→ Arrangement of symbols is imp

Alphabet: (set of symbols)

Alphabet: $\{s_1, s_2 \ldots s_n\}$

n = Vocabulary

↓

n / length of set.

organization of individual

→ Sequence of symbols → message produced by source using the alphabet
↳ represents info.

→ The frequency of occurence of symbol is also called probability.

each symbol i has probability P_i. dist P_i

Entropy $H = \sum P_i \log_2\left(\frac{1}{P_i}\right) = -\sum P_i \log_2 P_i$

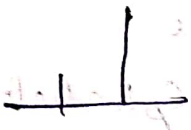
$\log_2\left(\frac{1}{P_i}\right)$ → Self-information

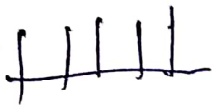
$-\log_2\left(\frac{1}{P_i}\right)$ = no of bits of information contained in symbol to send that message.

→ Entropy becomes weighted avg of info carried by each symbol, hence avg Symbol length.

→ More occuring symbols → more probability → More short words

→ Uniform probability distribution → No compression


More peaked distribution - Entropy will be low.


Flat

3/3/22

120 page in text book

from (110)

Bouncing ball ass.

Entropy

Avg no. of bits required to represent each symbol

3/3/22

→ Since - entropy indicates content in an information source s, it leads to a family of coding methods

→ entropy coding methods

→ (VLC) is one-best-known method

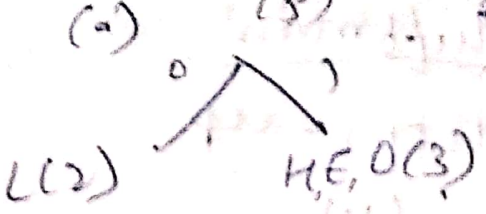
→ Shannon-Fano algo. (top-down)

→ Huffman coding - variable length encoding

Entropy encoding algorithms

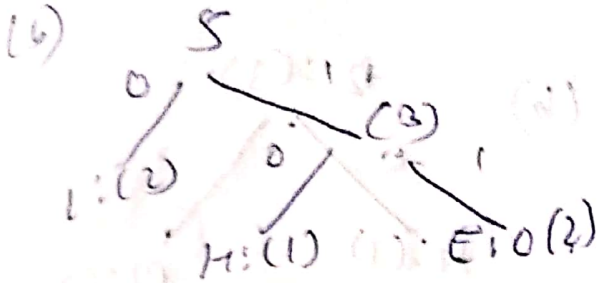
for lossless data compression

LHEO - 2 1 1

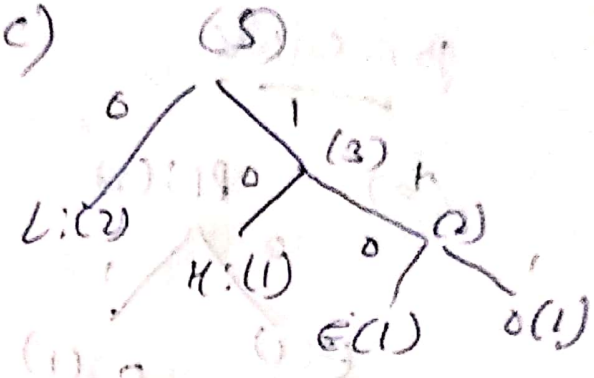


Shannon-Fano Algo

(b)



c)



Left - 0

Right - 1

Symbol	Count	$\log_2 \frac{1}{P_i}$	Code	No. of bits used
L	2	1.32	0	2
H	1	2.32	10	2
E	1	2.32	110	3
O	1	2.32	111	3
				10

Total no of bits

$$\eta = P_L \log_2 \frac{1}{P_L} + P_H \log_2 \frac{1}{P_H} + P_E \log_2 \frac{1}{P_E} + \dots \log_2 \frac{1}{P_O}$$

$$= 0.4 * 1.32 + 02$$

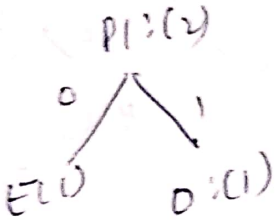
8/3/22 Huffman

→ ~~Top~~ Bottom - up.

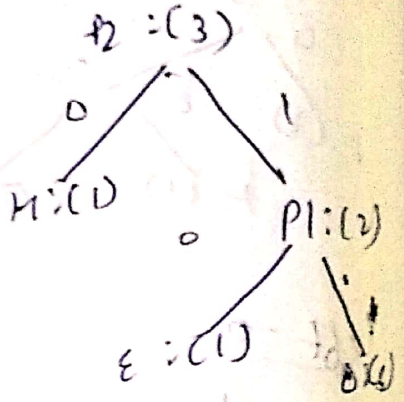
→ Sort the list.

Procedure

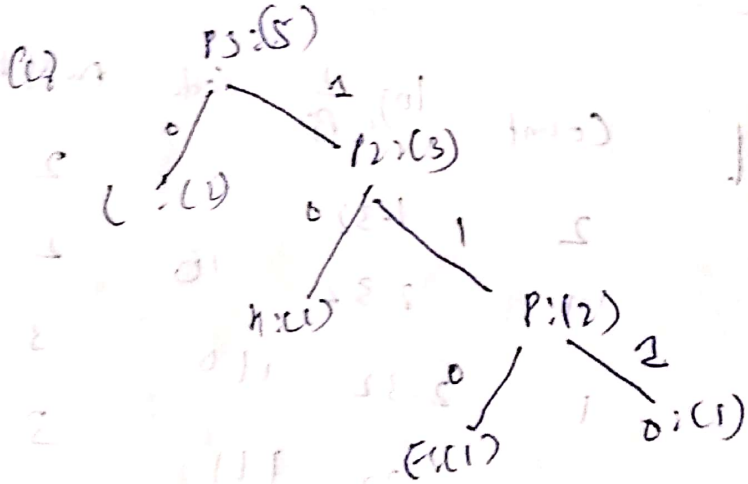
(a)



(b)



(c)



L : 0

H : 10

E : 110

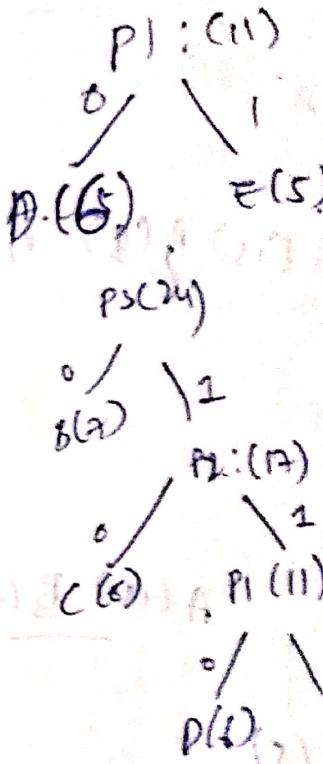
D : 111

Average = $\frac{\text{bits}}{\text{No of symbols}}$

= $\frac{(2+2+3+3)}{5}$?

A(15) B(7)

Huffman :



A → 0 (15)

B → 10 =

C → 110 =

D → 1110

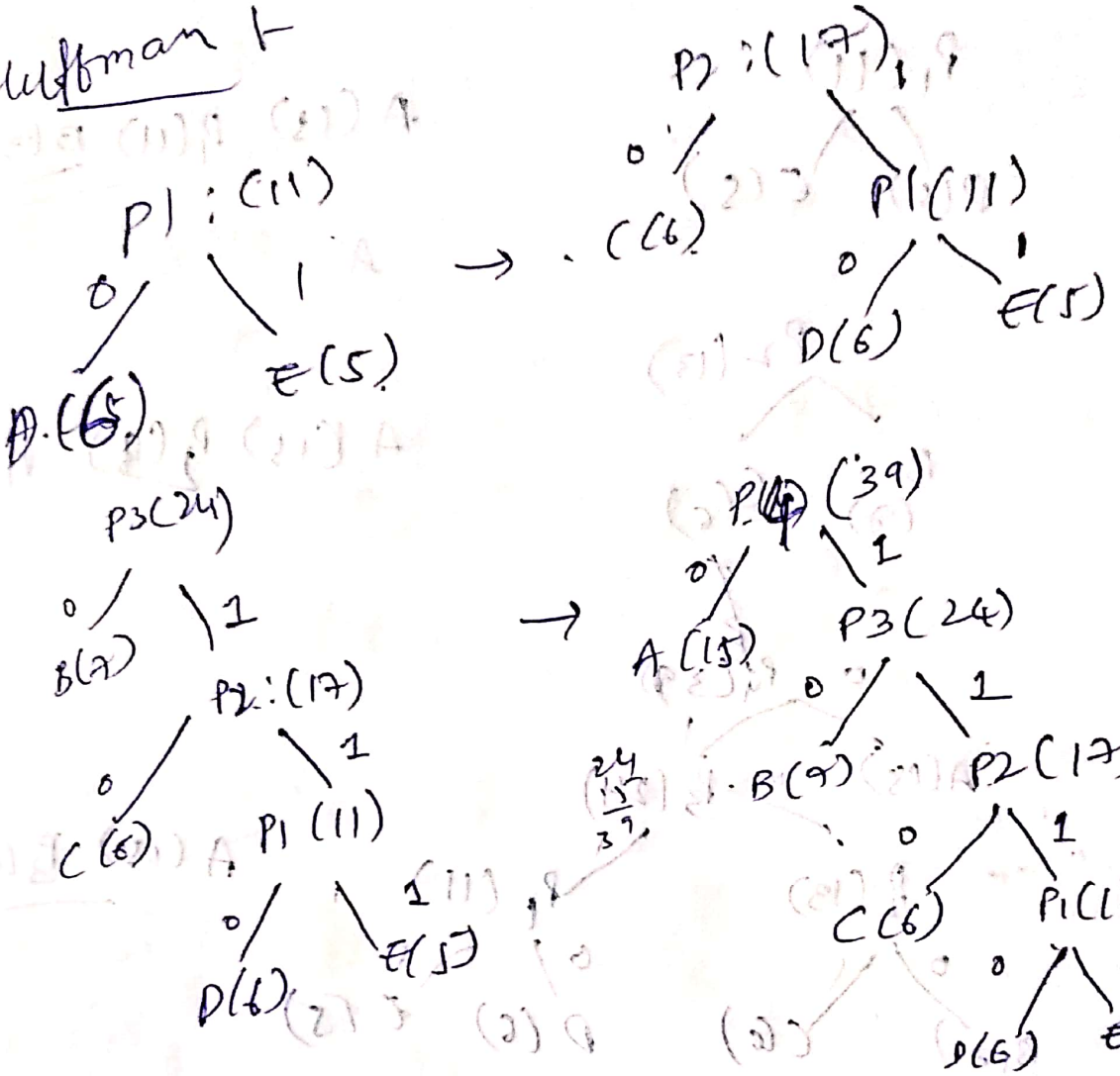
E → 1111 ?

Avg = ~~Avg~~ / length of

(X)

A(15) B(7) C(6) D(6) E(5)

Huffman :-



A → 0 (15) = 15

B → 10 = 2(7) = 14

C → 110 = 3(6) = 18

D → 1110 = 4(6) = 24

E → 1111 = 4(5) = 20

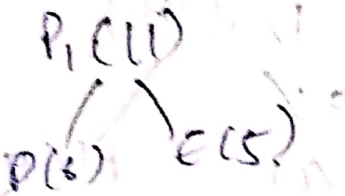
91/59 = 1.54 = 2.33

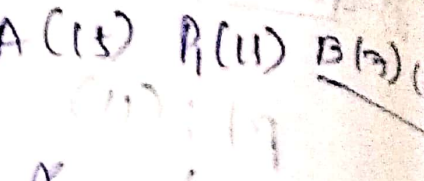
Avg = ~~freq~~ / length of word.

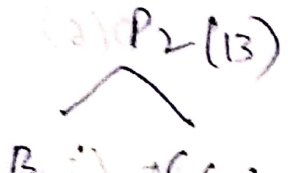
(X)

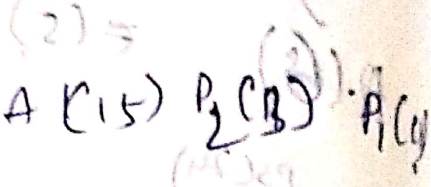
A (15) B (7) C (6) D (6) E (5)

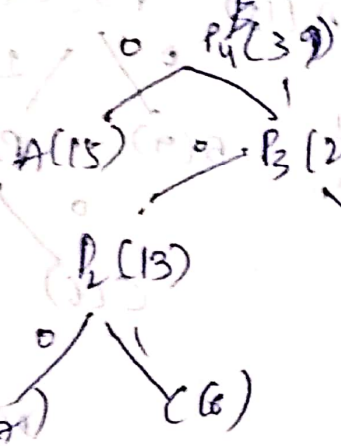
Huffman

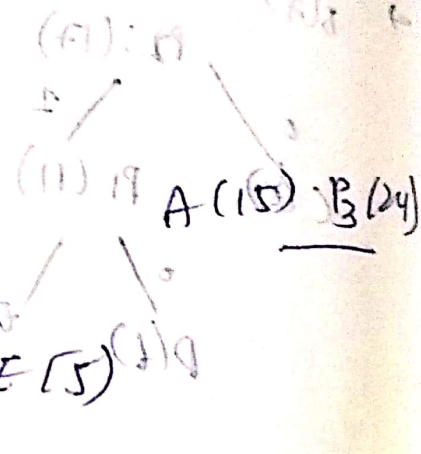












A → 0 = 1 (15) = 15

B → 100 = 3 (7) = 21

C → 101 = 3 (6) = 18

D → 110 = 3 (6) = 18

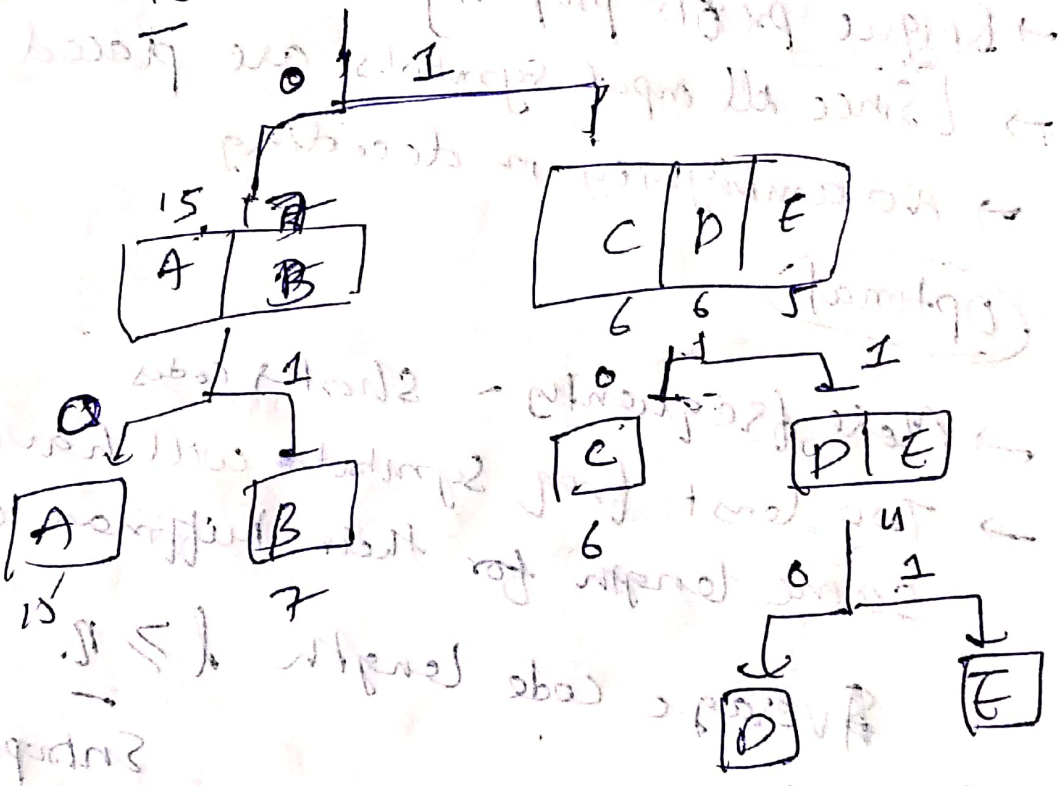
E → 111 = 3 (5) = 15

Shannon - Fano

	A	B	C	D	E
Prob	0.22	0.28	0.15	0.30	0.05

8 | 3 | 22

	A	B	C	D	E
Count	15	7	6	6	5



A → 00

B → 01

C → 10

D → 110

E → 111

Symbol	Count	$\log_2 \frac{1}{P_i}$	Code	No of bits
A	15		00	30
B	7		01	14
C	6		10	12
D	6		110	18
E	5		111	15
				89

89/35 = 2.54

Huffman's Coding is better. When dealing with large no of sets (codes) (words) than Shannon Fano Algo.

- Huffman produces good compression results
  ↓ properties
  =
  → Unique prefix property
  → [Since all input symbols are placed at leaf
  → no ambiguity in decoding

Optimal

→ More frequently - Shorter codes
→ Two least freq symbols will have same length for their Huffman codes

Average code length $l \geq \eta$
Entropy

$\eta \leq l < \eta + 1$

=

A → 0.6
B → 0.3
C → 0.1

Average Entropy

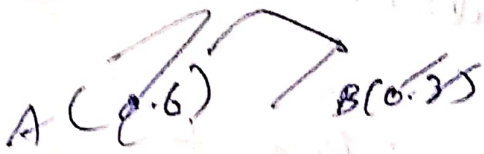
A → 1

B → 01

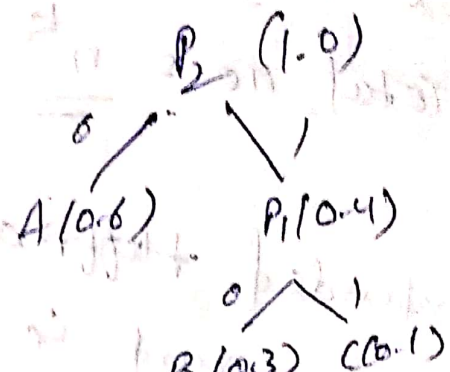
C → 00

A B C

0.6, 0.3, 0.1



A B C



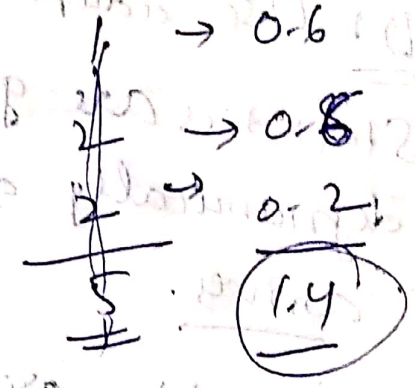
A : P1

0.6 0.4

A — 0 0.6

B — 10 0.3

C → 11 0.1



$$S = 0.6 \log\left(\frac{1}{P_1}\right)$$

$$= 0.6 \log_2\left(\frac{1}{0.6}\right) + 0.3 \log_2\left(\frac{1}{0.3}\right) + 0.1 \log_2\left(\frac{1}{0.1}\right)$$

10 का = 0.1

$$= 0.44 + 0.52 + 0.33$$

$$= 1.29$$

$n = 3$

$k = 2$ groups

Total no. = n^k

Difficult to draw

Problem size of tree is increased.

Extended Huffman bitrate does not into found & results in modest improvement

Adaptive Huffman Coding

→ Requires prior statistical knowledge

→ Solⁿ: Use adaptive compression algo

→ Statistics are gathered and updated dynamically as the datastream arrive

→ probabilities are no longer based on prior knowledge but on actual data

10/3/22 (Tutorial-2)

In anaconda

conda install -c conda-forge pyglet

10/3/22

Adaptive Huffman Coding

Encoder

```
Initial_code();
while not EOF
{
  get(c);
  encode(c);
  update_tree();
}
```

Decoder

```
initial_code();
while not EOF
{
  decode(c);
  output(c);
  update_tree();
}
```

Initial-code :- Some initial agreed-upon codes

update_tree :- Increases freq counts
Updates configuration of tree

Sibling-property :- nodes are arranged in non-increasing counts. [from left to right, bottom to top].

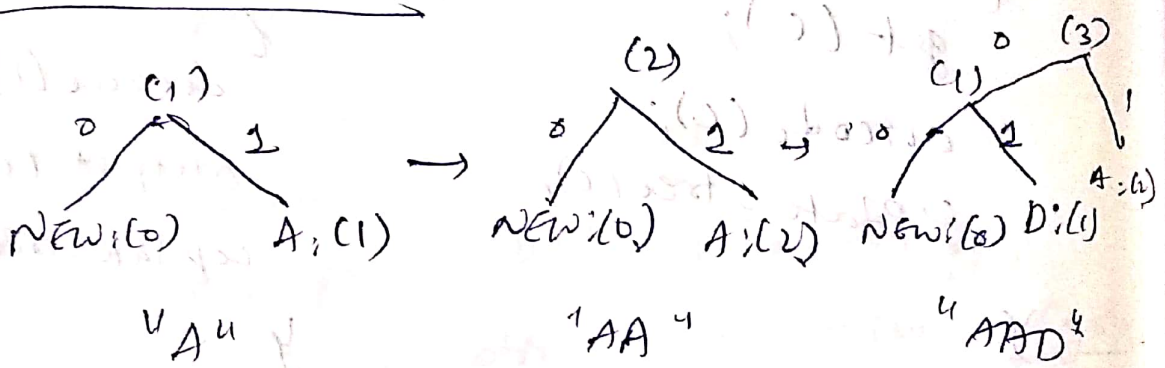
→ Encoder & Decoder must have same initial codes & update_tree routines

Rearranging :- Swap with leaf node [if current is leaf]
Swap with entire subtree [if current is not leaf]

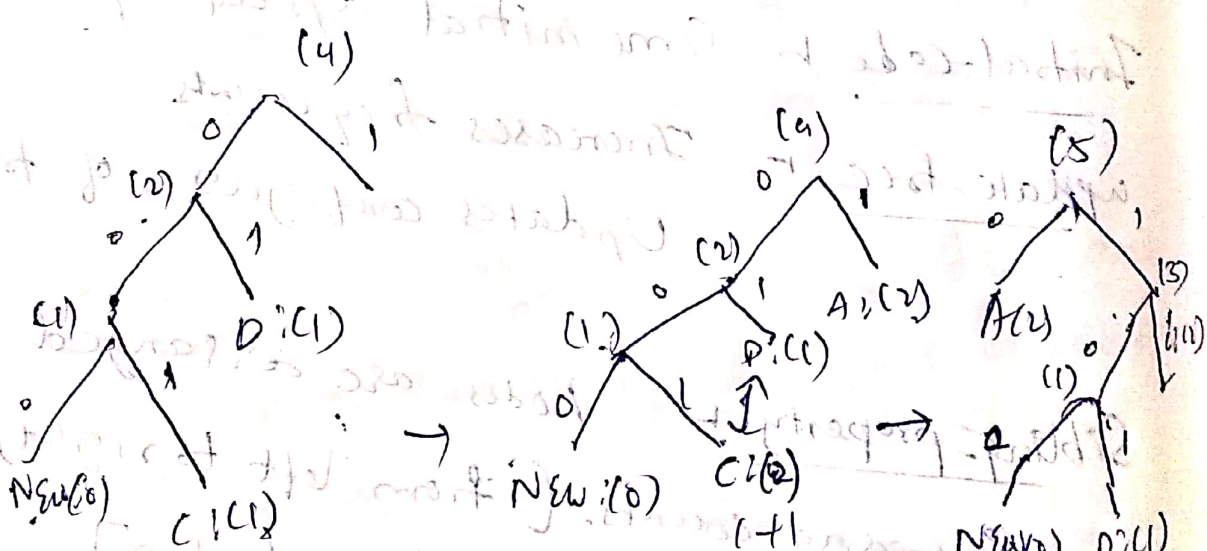
Building Adaptive Huffman Tree

→ new time a symbol is added new is preceded with symbol with count = 0.

For 'AADCCPD →

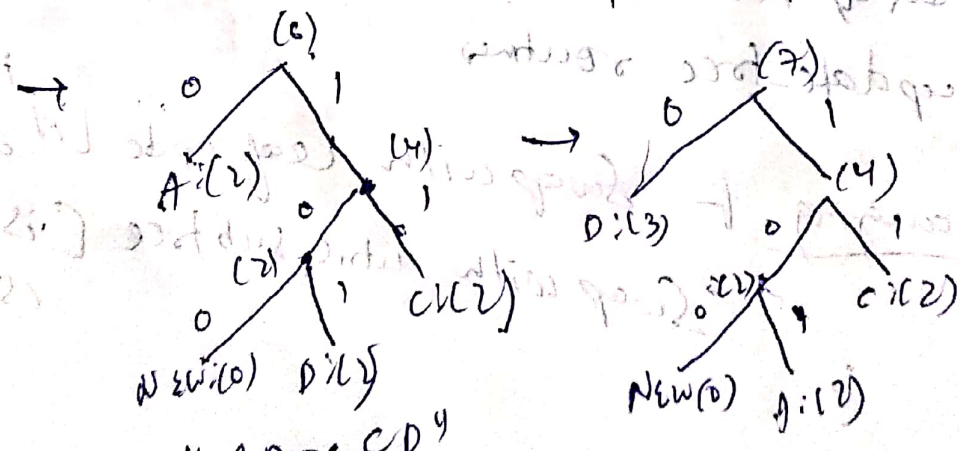


"A" "AA" "AAD"



"AADC" "AADCC"

(step)



"AADCCD"

Arithmetic

// (10/3) x2

(0 2 Pi)

BALL	Prob	CD	low	high
A	0.4	0.4	0	0.4
A	0.4	0.8	0.4	0.8
B	0.4	1.0	0.8	1.0
L	0.2			

BALL

l + range x low Probability of A

=

Two symbols. 3 A, B, C ——— P(C) = 0.25

P(A) = 0.5 P(B) = 0.25

Arithmetic code for BACA ;

14/8/22

ffmpeg
moviepy

from moviepy.editor import *
clip = VideoFileClip("same-mp4-file.mp4")

→ subclip – for specific size of clip
→ moviepy measures time in seconds

clip = VideoFileClip("sample-mp4-file.mp4")
.subclip(56, 66)

≡ AudioClips is moviepy work much the same as VideoClips. We create a new AudioClip in some way as VideoFileClip

= • set_audio fn replaces a video clip's audio with a new audio clip.

• save_frame("frame2.png", t=2)

- iterate()

BTP TP - 14/8/22

- magick wizard: wizard.jpg
  magick wizard.jpg win:

Convolution Neural Network - Good for image classification

LSTM - Long short term memory network - Good for speech recognition

Neuron - It's a thing that holds number

Every repo will have a new config & data files in Solr.

→ Solr Reference Guide [PPT → URL → code] Install.

Redis – Remote directory Server

→ Used as a database, Cache & message broker

→ Key-value store provides ability to store some data called value.

Main Adv :- [Master - Slave Arch]

→ Holds D.B entirely in memory. [RAM] = primary

→ Rich data types.

→ High speed :-

→ [multiple values in a single command]

→ Master slave Replication

Sharding :

– Easy to distribute dataset across multiple Redis instances

Solr & Redis - localhost : 8983 — solr

— 6379 - redis

Solr start

solr stop -all

15/3/22

LZW compression algo (Adaptive

S = next input character

S = A

C = B

Algo [Encoding]

BEGIN

S = next input character

While not EOF,

{ C = next input char;

if (present) then do : { if S+C exists in dic

S = S+C;

else

{

output the code for S;

add string S+C to the dictionary with a new code

S = C;

}

output code for S;

END

A - 1

B - 2

C - 3

(14)

// A B A B B A B C A B A B B A - LZW compression algo.

S	C	output	code	String
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
ABA	B			
ABB	A	6	11	ABBA
A	EOF	1		

→ 1 2 4 5 2 3 4 6 1 — (9)

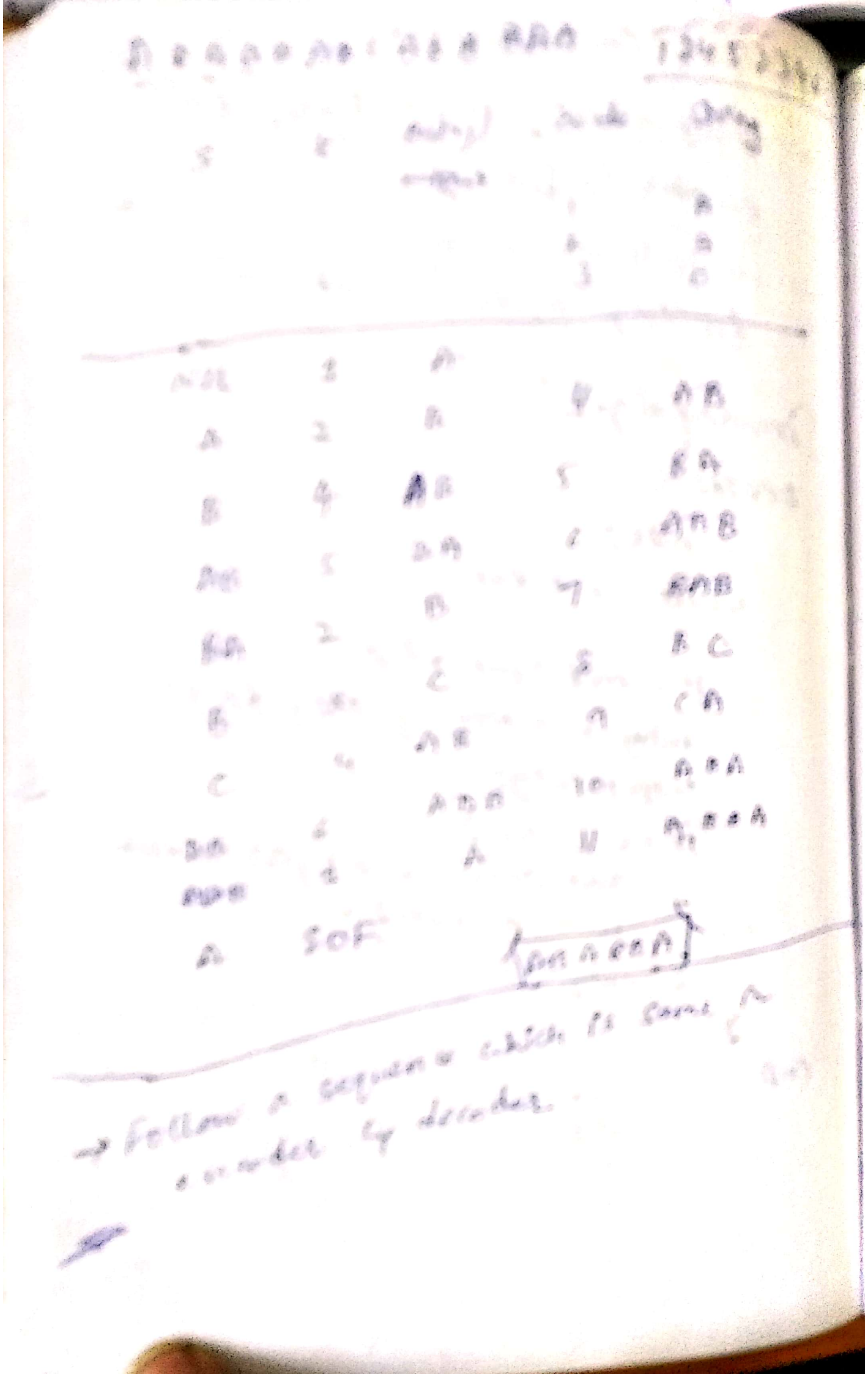
14:9 [Compression

→ Independently built tables for encoder & decoder.

→ We will achieve good compression ratio for large bytes [hundreds long]

[Decoding algo].

```
BEGIN
    S = NIL;
    while not EOF
    {
        k = next input code;
        entry = dictionary entry for k;
        output entry;
        if (S != NIL)
            add string S + entry[0] to dict
            with a new code;
        S = entry;
    }
END
```



- Adaptively updating the dictionaries, encoder is sometimes ahead of decoder.

Adaptive - Alg changes behaviour based on info available at that time.

Lossless - Huffman
- Shannon-Fano } Entropy-coding techniques
- Arithmetic

- Dictionary-based

=

lossy compression

Transform coding -

→ Performing a mathematical transform on input signal that results in a diff. signal

→ Transform coding are not lossy, but employ quantization after a transform

- Freq transforms - Discrete Fourier Transforms, Hadamard transforms, Lapped trans——

- Statistical transforms - Karhunen-Loeve transforms

- Wavelet - Trans to multi-resolution freq representation

rgb - 512x512

Good compression techniques have Hybrid of both lossy & lossless

Image compression

→ Irrelancy reduction
→ Redundancy reduction

Lossless
↳ TIFF, BMP
→ LZW, GIF, PNG
→ Prediction based JPEG
— Statistical based

Lossy
↳ Spatial Domain quantization
↳ Transform coding
↳ Sub band
↳ Fractional-based

Image is a 2-D signal.

Image $f(i,j)$ ⟶ Image $F(u,v)$

DCT (Discrete Cosine Transform)

→ Decorrelation of input signal in a data-independent manner.

$$F(u,v) = \frac{2C(u)C(v)}{\sqrt{MN}} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \cos\frac{(2i+1)u\pi}{2M} \cos\frac{(2j+1)v\pi}{2N} f(i,j)$$

17/3/22

JPEG Compression pipeline

Step 1: YCrCb format conversion. [YUV]

Step 2: To 4:2:0 subsampling.

Step 3: To 8×8 grid size.

Step 4: Each 8×8 block undergoes a DCT (Discrete Cosine Transform) Transformation.

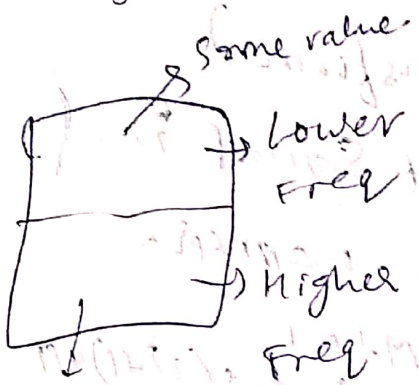
Step 5: Quantization

Using $F_Q(u,v) = \left[\frac{F(u,v)}{Q(u,v)}\right]$

$Q(u,v)$ from quantization table

Lower freq → Quantization is low.

Higher freq → Qu — is high



Step 6 :- Quantized Co-effecients $FQ(u,v)$ are then encoded into an intermediatary pattern.

Step 7 :- DCT blocks.

Step 8 :- Intermediary representation are then entropy coded using codes supplied by JPEG organization.

19/9/22

Mid prep :-

Step 1 :- In DCT block, first component is DC compon remaining all are AC comp

29/3/22 DC Co-effecients - Diff pulse code modul(D

AC Co-effec — Scanned in zigzag order

Intermediary Symbol	Binary repr [Symbol-1]	Binary rep [Symbol-2]

→ Blocky artifacts occur due to quantization at high bit rates [more compression ratio].

2 bits per pixel — Good

0.5 — Bad

0.15 — very Bad

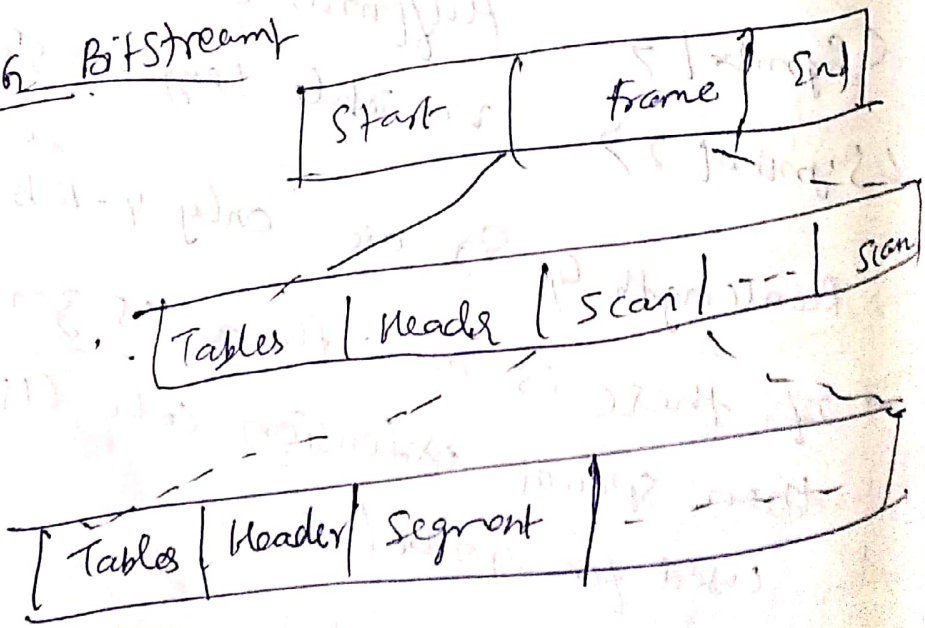
29/3/22

Drawbacks :-

→. Game, some animation, video. Consider multimedia

↳ Project ideas

JPEG Bitstream

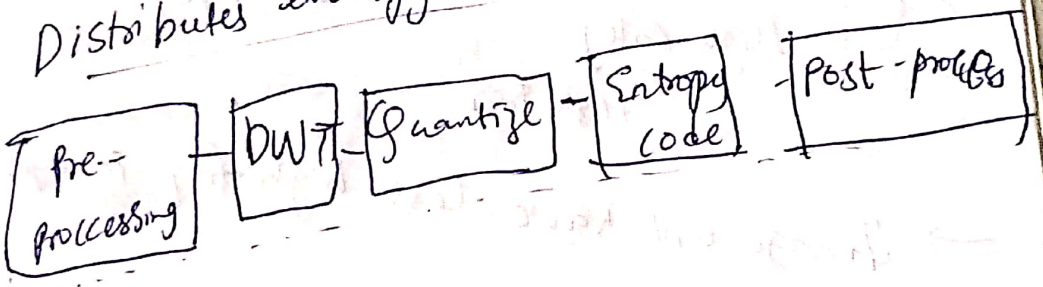


Drawbacks of JPEG

→ Poor - Low bit-rate compressions
→ Lossy & Lossless compression
→ Random access of the bit stream
→ No single compression architecture
→ Transmission is noisy environment is difficult

JPEG 2000

→ Makes use of DWT (Discrete wavelet Transform) to compress images
→ Distributes energy among freq co-efficients



1) Pre-processing step :-

1) Tiling :-
Partitioning into rectangular blocks (non-overlapping)

2) YCrCb conversion process :-

3) Level Offsetting process :-
Shifting the DC levels - Subtracting a const from each pixel value.

7) Discrete Wavelet Transform

→ Represent a signal with good resolution in both time & frequency, by splits
As called wavelets

→ High & low freq parts of image are independently processed.

→ Image is filtered along x-dimension

→ Rate at which pixel values are changed — Frequency
↳ High Rate → High freq

→ Image will have less high freq images

Original freq → Filter rows → Down Sample columns by 2 → Filter columns → Down sample rows by 2

LL	LH
HL	HH

1 stage

→ JPEG 2000 - supports from 0 to 32 stages

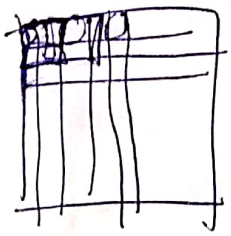
→

Stage1 LL	Stage1 LH
Stage1 HL	Stage1 HH

→

Stage2	Stage2	Stage1 LH
Stage2	Stage2	
Stage1 HL		Stage1 HH

31/3/22



3) Quantization :

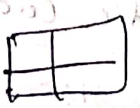
→ After quantization, co-efficients are quantized using scalar quantization.

→ Quantization reduces co-efficients in precision

4) Encoding :

→ Block-based encoding scheme known as Embedded Block coding with optimized Truncations (EBCOT).

=

Precinct :-  four code-blocks

→ Each sub-band is divided into rectangular blocks called precincts.

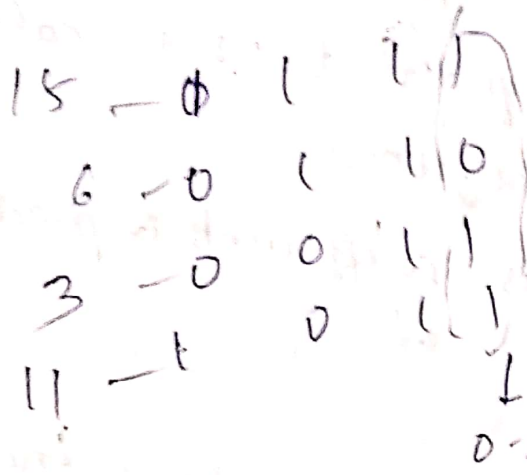
Code-blocks  → packet into further Over-lapping rectangles.

Packets-3 Spatially consistent rectangles comprise a packet

→ Each code block is encoded independently to the entropy encoder.

Entropy coding :- Bit-planes :-

→ Co-effecients of code block are seperated into bit-planes



Bit-plane

Coding passes → Bit-plane context (represent) → [context-based Arithmetic coding] → Compressed bit Stream

3- Coding passes :- [Arithmetic coding]

1) Significance propogation pass :-
   - coffecients that are significant are coded

2) Magnitude refinement pass :- (current bits of significant co-effecients are encoded)

3) Clean-up pass :-

5/4/22

Arithmetic coding :- Based on probabilities we approximate, the given string, then the last probability value is converted to binary value.

→ Once, the non-zero bit is encoded, the co-effecient becomes significant

→ Once significant, all subsequent bits are refered to as refinement bits insignificant still zero

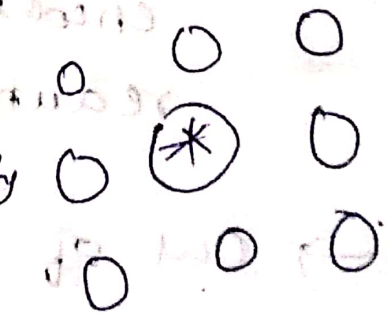
→ Most info is present in low freq sub-band

3 coding passes :-

(1) → Significane propog pass :-

Insignificant bits that have high probab of becoming significant are considered & encoded.

→ Neighbours decide probability of significance.



(2) Magnitude refinement pass :-

Significant co-effecients are refined by their bit representation

(3) Clean-up pass :-

→ Remaining co-efficients of bit plane are encoded as they have low prob of becoming significant.

=

MQ coder :-

Based on probability, probability changes with incoming bits encoded in binary.

- context-based
- Adaptive
- Binary arithmetic coding

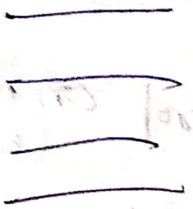
→ Context-based adaptive binary arithm coding (MQ coder)

→ JPEG2000 uses an efficient method for encoding uses MQ coder exploiting redundancy in bit-planes

→ Out of 8 neighbours, $2^8 = 256$ different possibilities, only 9 possibilities (context) are taken.

→ Sign encoding :- Only horizontal & vertical neighbours

Special mode (run mode) :-



} vertically vertically 4 samples have insignificant neighbours.

 → Each codeblock is independently coded, MQ-coder individual to each codeblock with diff probab-

Each codeblock will generate single arithmetic code.

→ 256x256 image
128x128 — level 1 decomposition
64x64 — level 2 decompo

→ Arithmetic coding of bitplane data is referred to as tier-1 (T1) coding.

→ In tier-2 coding, compressed data into blocks units as packets & layers.

5/4/22

JPEG 2000 Vs JPEG :

→ JPEG 2000 - Have control on resolution (high)

JPEG → Doesn't have control.

→ Region of Interest Encoding

→ Working with compressed images

→ High compression (low bitrates) with better quality

→ Error resilience (especially at robustness to bit errors when communicating or storage devices are correlated)

→ Support for tiling.

→ JPEG only upto 64k × 64k.

→ JPEG 2000 more than 64k.

Video compression :

→ Time ordered sequence of frames, i.e. images

→ Images are having highly redundant data i.e., adjacent pixels are highly correlated

→ Exploit spatial & spectral redundancy

M-JPEG :-

→ JPEG compression for each image or frame [Total video].

7/4/22

→ Encode the frame difference

Macro Block :- Compression method works on a block of 16x16 pixels called macro blocks.

- I (Intra) frame
  → Independent frames
- P (Predictive) or Inter frame
  → Not independent

Reference frame

Target Frame

→ direction of motion vector

P-frame :

→ MPEG-1 Evolution :

→ MPEG-1 supports only non-interlaced video.
- 4:2:0 chroma subsampling

→ Introduces a new B-frame
(Bi-directional predictive coded).

→ prediction is from previous frame, Forward prediction
" next frame, Backward prediction

→ Reduces frame difference, to achieve higher compression

→ The display order contains I, B, P frames.

Display order - I B B P B B P B B I

Coding & transmission order - I P B B P B B I B B

B → I, P
P → I, P
} dependent

I → Independent

Questions in endsem

B frames need to be buffered causing delay

MPEG

M - Interval between P & preceding I & P frame.

N - In between I-frames

(CD's)

Issues :-

→ Buffering issue.

→ Stores & plays video at low-bit rate. (1.5 MBPS)

MPEG-2 :-

→ Adopted in DVD's.

→ Higher bit rates (4MBPS)

→ MPEG-2 also use I, P, B frames but with half pixel approximation (in Interlaced video also only)

→ Support for variety of packet formats with error-correction capability:

→ Scalable encoding:

→ Input layer is broken into 2 layers, low freq & high freq layer. Both layers are coded independently, then multiplexed into bit stream.

MPEG-7 (later)

7/4/22

MPEG-4:

→ Version 10 of visual part is known as MPEG-4 AVC (Advanced visual encoding)

→ Supports many bandwidth ranges.

→ Supports both progressive & interlaced video encoding.

→ Standard is object based encoding

→ 25% better than MPEG-2

→ Temporal scalability

4th Unit.

Done

Multimedia Communication Cy

Module 3:

[Unit-5] Networking

OSI

→ Physical – Cables (10Base T, 100Base T).
→ Data link – (MAC, LLC – Ethernet:
media access control)
→ Network – IP (IPv4, IPv6)
→ Transport – TCP, UDP
→ Session – OSI
→ presentation – FTP, Multipurpose IME
→ Application – ~~FTP, HTTP, Telnet, SMTP,~~
~~ACP~~ DNS, DHCP, FTP

OSI – 7 layers
TCP/IP – 5 layers. Application, Transport
app presen session

→ Volume Cy Continuous nature of multm data
→ Realtime Cy Interactiveness of multimedia data
→ Rate fluctuations.

9/4/22

AR Application

Manufacturing,
Automated
Support
Maintenance
Customer Experience
Entertainment

12/4/22

- SIP (Session Initiation Protocol)
  - → Application-layer protocol
  - → Voice over IP
  - → Proxy, Redirect, Location → Servers
- SDP (Session Description Protocol)
  - Internet Telephony
  - → Provides great flexibility
  - → Uses packet switching
  - → Various degrees of QoS is achieved.

→ Network protocol structure for internet telephony.

→ Multimedia conferencing

Types :- 1) Point-to-point conferencing (2 people)

2) Multi-point video conferencing (2 or more)

→ VoIP makes video conferencing possible.

→ VoIP relies on special algorithms called codecs (coder-decoder)

- Data compression (coding)
- Data transfer & decompression (decoding)

Features of video conferencing :-

- Screen sharing
- Chat Box
- → File sharing
- → Video call recording.
- → Noise cancellation

- Video on Demand (VOD) :-

→ Allows users to access videos without a traditional video.

3 advantages :- → watch at any time.
→ Control what they watch
→ Use media controls.

VOD makes money :

Different types of VOD models To choose

1) TVOD or Transactional VOD :-
   → Pay-per-view model [Youtube movies]

2) SVOD or Subscription VOD :
   - Netflix, Amazon Prime.

3) AVOD or (Advertising Video on Demand)
   → Need to watch advertisements

Key Issues of VOD :

→ Content Acquisition
→ Bandwidth issues
→ Platform Accessibility
→ Rentals & Pricing